Querying Multiple Tables

```
/*Querying Multiple Tables
Joins - Inner Join*/


desc customer;


desc address;


/*Let’s say you want to retrieve the first and last names of each customer, along with their street address. Your query will therefore need to retrieve the customer.first_name, customer.last_name, and address.address columns*/


SELECT
    customer.first_name, customer.last_name, address.address
FROM
    customer
        JOIN
    address;


SELECT
    c.first_name, c.last_name, a.address
FROM
    customer AS c
        JOIN
    address AS a;
```

/*Hmmm...there are only 599 customers and 603 rows in the address table, so how did the result set end up with 361,197 rows? Looking more closely, you can see that many of the customers seem to have the same street address. Because the query didn't specify how the two tables should be joined, the database server generated the **Cartesian product**, which is every permutation of the two tables (599 customers x 603 addresses = 361,197 permutations). This type of join is known as a **cross join***/

```
SELECT
    c.address_id,c.first_name, c.last_name, a.address
FROM
    customer AS c
        JOIN
    address AS a ON c.address_id = a.address_id;
```

/*If a value exists for the address_id column in one table but not the other, then the join fails for the rows containing that value, and those rows are excluded from the result set. This type of join is known as an inner join, and it is the most commonly used type of join. To clarify, if a row in the customer table has the value 999 in the address_id column and there's no row in the address table with a value of 999 in the address_id column, then that customer row would not be included in the result set. If you want to include all rows from one table or the other regardless of whether a match exists, you need to specify an outer join*/

```
SELECT
    c.first_name, c.last_name, a.address
FROM
    customer AS c
        INNER JOIN
    address AS a ON c.address_id = a.address_id;
```

/*If you do not specify the type of join, then the server will do an inner join by default for inner join we can use the sub clause using instead of on since two columns used to join the two tables are identical.*/

```
SELECT
    c.first_name, c.last_name, a.address
FROM
    customer AS c
        INNER JOIN
    address AS a USING (address_id);
```

/*Write a query, which returns only those customers whose postal code is 52137*/

```
SELECT
    c.first_name, c.last_name, a.address
FROM
    customer AS c
        INNER JOIN
    address AS a USING (address_id)
WHERE
    a.postal_code = '52137';
```

```
SELECT
    c.first_name, c.last_name, a.address
FROM
    customer AS c
        INNER JOIN
    address AS a ON c.address_id = a.address_id
WHERE
    a.postal_code = '52137';
```

Joining three or more tables

/*Joining three or more tables Joining three tables is similar to joining two tables, but with one slight wrinkle. With a two-table join, there are two tables and one join type in the from clause, and a single on subclause to define how the tables are joined. With a three-table join, there are three tables and two join types in the from clause, and two on subclauses.*/

```
desc customer;

desc address;

desc city;

SELECT
    c.first_name, c.last_name, city.city
FROM
    customer AS c
        INNER JOIN
    address AS a USING (address_id)
        INNER JOIN
    city USING (city_id);

SELECT
    c.first_name, c.last_name, city.city
FROM
    customer AS c
        INNER JOIN
    address AS a ON c.address_id = a.address_id
        INNER JOIN
    city ON a.city_id = city.city_id;
```

```
/*At first glance, it might seem like the order in which the tables
appear in the from clause is important, but if you switch the table
order, you will get the exact same results. But All three of these
variations return the same results:*/

SELECT

    c.first_name, c.last_name, ct.city

FROM

    customer c

        INNER JOIN

    address a ON c.address_id = a.address_id

        INNER JOIN

    city ct ON a.city_id = ct.city_id;


SELECT

    c.first_name, c.last_name, ct.city

FROM

    city ct

        INNER JOIN

    address a ON a.city_id = ct.city_id

        INNER JOIN

    customer c ON c.address_id = a.address_id;


SELECT

    c.first_name, c.last_name, ct.city

FROM

    address a

        INNER JOIN

    city ct ON a.city_id = ct.city_id

        INNER JOIN

    customer c ON c.address_id = a.address_id

ORDER BY city;
```

/*Using Subqueries as Tables*/

```
SELECT
        a.address_id, a.address, c.city
    FROM
        address AS a
    INNER JOIN city AS c USING (city_id)
    WHERE
        a.district = 'California';

SELECT
    c.first_name, c.last_name, ad.address, ad.city
FROM
    customer AS c
        INNER JOIN
    (SELECT
        a.address_id, a.address, c.city
    FROM
        address AS a
    INNER JOIN city AS c USING (city_id)
    WHERE
        a.district = 'California') ad USING (address_id);
```

```
/*If you are joining multiple tables, you might find that you need to
join the same table more than once.*/


desc film;


SELECT
    f.title
FROM
    film AS f
        INNER JOIN
    film_actor USING (film_id)
        INNER JOIN
    actor AS a USING (actor_id)
WHERE
    ((a.first_name = 'CATE'
        AND a.last_name = 'MCQUEEN')
        OR (a.first_name = 'CUBA'
        AND a.last_name = 'BIRCH'));
```

```
 SELECT
    f.title
FROM
    film AS f
        INNER JOIN
    film_actor AS fa1 on f.film_id = fa1.film_id
        INNER JOIN
    actor AS a1 ON fa1.actor_id = a1.actor_id
        INNER JOIN
    film_actor AS fa2 ON f.film_id = fa2.film_id
        INNER JOIN
    actor AS a2 ON fa2.actor_id = a2.actor_id
WHERE
    ((a1.first_name = 'CATE'
        AND a1.last_name = 'MCQUEEN')
        AND (a2.first_name = 'CUBA'
        AND a2.last_name = 'BIRCH'));
```

/*Self-Joins

Not only can you include the same table more than once in the same query, but you can actually join a table to itself*/

/* Exercise - 1

Fill in the blanks (denoted by <#>) for the following query to obtain the results that follow:

```
mysql> SELECT c.first_name, c.last_name, a.address, ct.city
-> FROM customer c
-> INNER JOIN address <1>
-> ON c.address_id = a.address_id
-> INNER JOIN city ct
-> ON a.city_id = <2>
-> WHERE a.district = 'California';
+------------+-----------+-----------------------+----------------+
| first_name | last_name | address               | city           |
+------------+-----------+-----------------------+----------------+
| PATRICIA   | JOHNSON   | 1121 Loja Avenue      | San Bernardino |
| BETTY      | WHITE     | 770 Bydgoszcz Avenue  | Citrus Heights |
| ALICE      | STEWART   | 1135 Izumisano Parkway | Fontana        |
| ROSA       | REYNOLDS  | 793 Cam Ranh Avenue   | Lancaster      |
| RENEE      | LANE      | 533 al-Ayn Boulevard  | Compton        |
| KRISTIN    | JOHNSTON  | 226 Brest Manor       | Sunnyvale      |
| CASSANDRA  | WALTERS   | 920 Kumbakonam Loop   | Salinas        |
| JACOB      | LANCE     | 1866 al-Qatif Avenue  | El Monte       |
| RENE       | MCALISTER | 1895 Zhezqazghan Drive | Garden Grove   |
+------------+-----------+-----------------------+----------------+

Ans - a & ct.city_id */
```

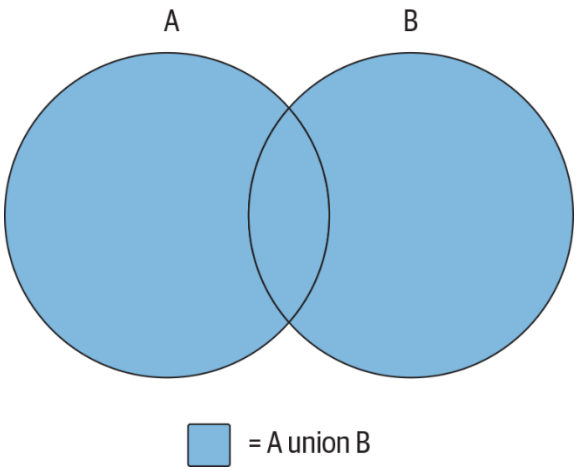
/*Exercise - 2 Write a query that returns the title of every film in which an actor with the first name JOHN appeared.*/

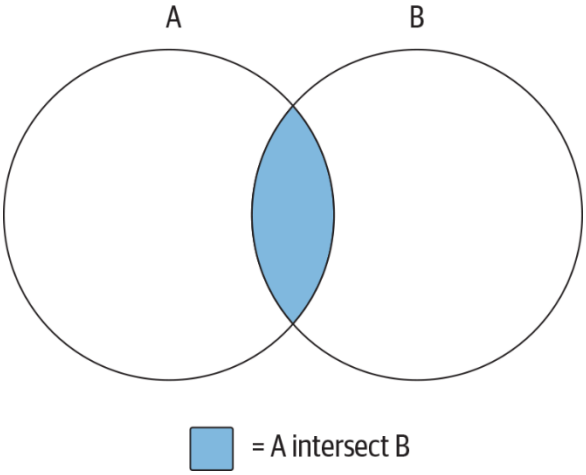
```
desc actor;
desc film;
desc film_actor;

SELECT
    f.title
FROM
    film AS f
        INNER JOIN
    film_actor AS fa ON f.film_id = fa.film_id
        INNER JOIN
    actor AS a ON fa.actor_id = a.actor_id
WHERE
    a.first_name = 'John';
```

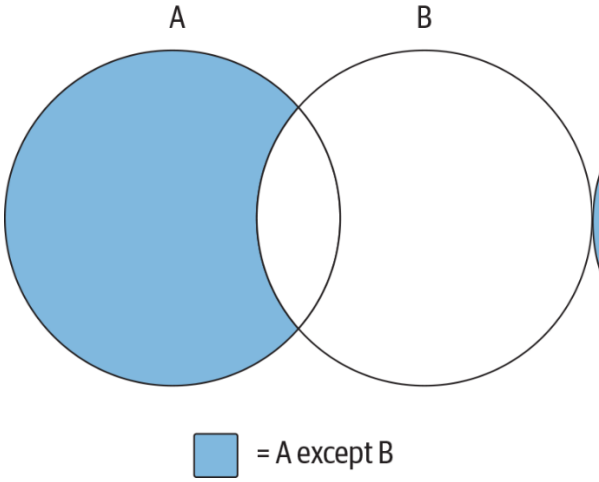
/*Exercise - 3 Construct a query that returns all addresses that are in the same city. You will need to join the address table to itself, and each row should include two different addresses.*/

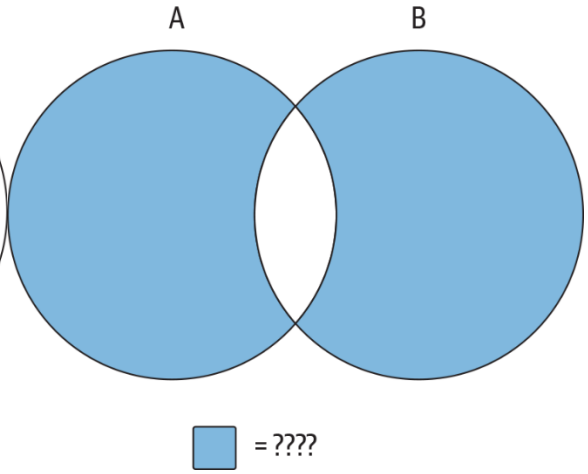
```
desc address;
desc city;

SELECT
    a1.address, a2.address, a1.city_id
FROM
    address AS a1
        INNER JOIN
    address AS a2
WHERE
    a1.city_id = a2.city_id
        AND a1.address_id <> a2.address_id;
```

SETs










```
(A union B) except (A intersect B)
```

```
desc customer;


desc city;


/*when performing set operations on two data sets, the following guidelines must apply:
• Both data sets must have the same number of columns.
• The data types of each column across the two data sets must be the same (or the server must be able to convert one to the other).
The SQL language includes three set operators that allow you to perform each of the various set operations
```

The Union Operator*/

```
SELECT
    c.first_name, c.last_name
FROM
    customer AS c
UNION ALL SELECT
    a.first_name, a.last_name
FROM
    actor AS a;
```

/*union all operator doesn’t remove duplicates*/

```
SELECT
    a.first_name, a.last_name
FROM
    actor AS a
UNION ALL SELECT
    a.first_name, a.last_name
FROM
    actor AS a;
```

```
SELECT
    c.first_name, c.last_name
FROM
    customer c
WHERE
    c.first_name LIKE 'J%'
        AND c.last_name LIKE 'D%'
UNION ALL SELECT
    a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'J%'
        AND a.last_name LIKE 'D%';
```

/*If you would like your combined table to exclude duplicate rows, you need to use the union operator instead of union all*/

```
SELECT
    c.first_name, c.last_name
FROM
    customer c
WHERE
    c.first_name LIKE 'J%'
        AND c.last_name LIKE 'D%'
UNION
SELECT
      a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'J%'
        AND a.last_name LIKE 'D%';
```

/*Intersect Operrator & The except Operator - Not there in MySQl but available in Oracle or sql server 2008 */

/*Set Operation Rules

When specifying column names in the order by clause, you will need to choose from the column names in the first query of the compound query. Frequently, the column names are the same for both queries in a compound query, but this does not need to be the case, as demonstrated by the following*/

```
SELECT
    c.first_name, c.last_name
FROM
    customer c
WHERE
    c.first_name LIKE 'J%'
        AND c.last_name LIKE 'D%'
UNION ALL SELECT
    a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'J%'
        AND a.last_name LIKE 'D%'
ORDER BY first_name , last_name;
```

```
SELECT
    c.first_name as fname, c.last_name as lname
FROM
    customer c
WHERE
    c.first_name LIKE 'J%'
        AND c.last_name LIKE 'D%'
UNION ALL SELECT
    a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'J%'
        AND a.last_name LIKE 'D%'
ORDER BY first_name , last_name; --This won't work
SELECT
    c.first_name as fname, c.last_name as lname
FROM
    customer c
WHERE
    c.first_name LIKE 'J%'
        AND c.last_name LIKE 'D%'
UNION ALL SELECT
    a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'J%'
        AND a.last_name LIKE 'D%'
ORDER BY fname , lname;
```

/*Set Operation Precedence*/

```
SELECT
    a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'J%'
        AND a.last_name LIKE 'D%'
UNION ALL SELECT
    a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'M%'
        AND a.last_name LIKE 'T%'
UNION SELECT
    c.first_name, c.last_name
FROM
    customer c
WHERE
    c.first_name LIKE 'J%'
        AND c.last_name LIKE 'D%';
```

/*Same compound query with the set operators reversed*/

```
SELECT
    a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'J%'
        AND a.last_name LIKE 'D%'
UNION SELECT
    a.first_name, a.last_name
FROM
    actor a
WHERE
    a.first_name LIKE 'M%'
        AND a.last_name LIKE 'T%'
UNION ALL SELECT
    c.first_name, c.last_name
FROM
    customer c
WHERE
    c.first_name LIKE 'J%'
        AND c.last_name LIKE 'D%';
```

/*In general, compound queries containing three or more queries are evaluated in order from top to bottom*/

/*Exercise 1 -

If set A = {L M N O P} and set B = {P Q R S T}, what sets are generated by the following

operations?

- A union B - {L,M,N,O,P,Q,R,S,T}
- A union all B - {L,M,N,O,P,P,Q,R,S,T}
- A intersect B - {P}
- A except B - {L,M,N,O} */

/*Exercise 2 - Write a compound query that finds the first and last names of all actors and customers

whose last name starts with L.*/

```
SELECT
    first_name, last_name
FROM
    actor
WHERE
    last_name LIKE 'L%'
UNION SELECT
    first_name, last_name
FROM
    customer
WHERE
    last_name LIKE 'L%';
```

/*Exercise 3 - Sort the results from Exercise 2 by the last_name column.*/

```
SELECT
    first_name, last_name
FROM
    actor
WHERE
    last_name LIKE 'L%'
UNION SELECT
    first_name, last_name
FROM
    customer
WHERE
    last_name LIKE 'L%'
ORDER BY last_name;
```